

- 새 채팅
- 채팅 검색
- 라이브러리
- Codex
- Sora
- GPT
- Flux 이미지 생성 GPT
- 로나GPT
- [지피터스] 스타디장 지원서 제출
- PRD Builder
- 새 프로젝트
- AI 교육 담당자 채용
- 강의앱
- AI 튜터 로드맵 만들기
- 지피터스 AI 튜터
- 지피터스 AI 코치
- 더 보기

채팅

- 이력서 평가 루브릭 작성
- Linear 사용법 조사
- AI 응용 엔지니어 JD
- 영어 번역 요청
- AAE JD 작성 요청
- 1월 15일 이후 게시글
- AAE 이력서 검토 방법
- 프론트엔드 채용 정보
- 전문가 협업으로 JD 검토
- 과학적 채용 프로세스
- 이력서 스크리닝 분업 계획
- 직무명 비교 분석
- AI 엔지니어 역할 정리
- 회의록 정리 요청
- VOX 강의 피드백

jd-aae.md 파일

첨부된 파일에 묘사된 업무를 할 AI 응용 엔지니어를 채용하려해. 후보자의 이력서를 검토하기 위한 루브릭을 상세하게 작성해줘.

50s 동안 생각함

아래 루브릭은 첨부 JD의 역할/요건/산출물을 기준으로 설계했습니다.

## AI 응용 엔지니어 이력서 평가 루브릭 (Resume Rubric)

### 0) 한눈에 보는 사용법

- 평가 방식: 항목별 0-5점(레벨 앵커) × 가중치 → 가중 합산(100점 만점)
- 컷오프:
  - 80점 이상: 최종 면접 바로 진행
  - 65-79점: 실무자 스크리닝 면접(라이브 데모/포트폴리오 중심)
  - 50-64점: 보류(추가 정보 요청: 샘플 코드/데모)
  - 49점 이하: 불일치(피드백 메모 남기고 종료)
- 노크아웃(필수): Next.js/React/TS/Node 3년+ 실무, Postgres/Prisma 경험, REST/GraphQL 연동, AI 프롬프트/컨텍스트 기본 이해, 노코드/SaaS 연동 경험. 하나라도 전무하면 탈락.

### 1) 평가 항목·가중치(총 100점)

각 항목은 0-5점으로 채점합니다. 아래 레벨 앵커와 이력서에서 찾을 신호를 함께 확인하세요.

- 핵심 스택 적합도 (Next.js/React/TS/Node) — 20점**
  - 레벨 앵커:
    - 0: 경험 없음 / 학습 단계
    - 2: 토이/사이드 수준, 프로덕션 소규모
    - 4: 프로덕션 다수 릴리스, 성능/보안 고려 포함
    - 5: 복잡한 기능(권한/세션/피처플래그) 리드, 레거시 개선 리드
  - 이력서 신호: 대규모 사용자 대상 웹, SSR/ISR, 권한/세션 미들웨어, 피쳐 플래그, 성능 튜닝/모니터링
- 데이터-API 연동력 (PostgreSQL/Prisma/REST/GraphQL) — 15점**
  - 앵커: 0(없음)-5(스키마 표준화, 마이그레이션 전략, PII 분리/토큰화/보존정책 설계 경험)
  - 신호: Prisma 스키마 설계, 마이그레이션 운영, GraphQL/웹훅, API 보안(서명 검증), 캐시/큐 활용
- AI 응용 역량 (프롬프트/컨텍스트/제이닝/툴콜) — 15점**
  - 앵커: 0(키워드만 읽기)-5(역할/제약/포맷/예시 구조화, 함수-툴콜 설계, 가드레일/정책 적용)
  - 신호: 제이닝 설계, 컨텍스트 패키지/서브 에이전트, 실패 모드 대응, 파인튜닝/실시간 음성·영상 API 활용 사례
- 플랫폼 연동 실무 (Bettermode/PortOne/Cloudflare) — 10점**
  - 앵커: 0(없음)-5(SSO/역할 맵핑, 웹훅 보안 검증, 결제/정산/환불 플로우 운영, Workers/Queues 크론/재시도/데드 레터)
- 노코드·SaaS 조합과 빌드 vs 바잉 판단 — 10점**
  - 앵커: 0(무경험)-5(스크래핑/자동화 파이프라인 + 적절한 상용 도구 조합으로 리드타임 단축 입증)
  - 신호: Zapier/Make/내부 워크플로우, 비용/리스크 비교 의사결정 사례
- 관측성·실험 (로그/트레이싱/Eval/A/B/비용 모니터링) — 10점**
  - 앵커: 0(무관심)-5(실험 설계-수집-판정-롤아웃 루프 상시 운영)
  - 신호: 프롬프트/모델 Eval, 샘플사이즈 추정, 비용/성능 대시보드, 멀티버리언트 테스트
- 도메인 자동화 적합성 (그로스/세일즈/교육) — 10점**
  - 앵커: 0(무관)-5(ROI/ROAS/LTV/CAC, 퍼널/리텐션, CRM 세그/개인화, 교육 커리큘럼/평가 자동화 실적)
  - 신호: 인입 콘텐츠 다변화, 퍼널 운영/요약, 개인화 제안서 생성, 교육 자료/평가 파이프라인
- 테스트·신뢰성 (Vitest/Playwright/E2E) — 5점**
  - 앵커: 0(테스트 無)-5(계약/통합/E2E 체계 + CD에서 게이팅)
- 실행 속도·도구 활용 (“바이브 코딩”, AI 코딩 CLI) — 5점**
  - 앵커: 0(느림)-5(외부 AI 툴 적극 도입, 짧은 주기 배포, 최소 코드로 가치 검증)
- 문서화·운영 위생(SOP/Runbook/보안) — 5점**
  - 앵커: 0(없음)-5(SOP/Runbook/대시보드/알림/보안 체크리스트 일상화)

### 2) 가산/감점 규칙

- 가산(+최대 10%)

- 결제 시스템 프로덕션(PortOne 우대), 고가용성 운영, Cloudflare Workers/Queues 심화
- 실시간 음성/영상 생성 또는 파인튜닝 실제 배포 사례
- tRPC, Tailwind, Vitest/Playwright 숙련, 레거시 개선 리드 경험
- 스타트업 환경에서 0→1 구축 및 지표 개선(전환율, CAC↓ 등)

jd-aaa

- **감점(-최대 10%)**
  - 과도한 PoC 위주(프로덕션 전환 흔적 부족), 보안/PII 고려 전무, 지표·결과 미기재
  - "ChatGPT 써봄" 수준의 AI 표기만 있고 체계/관측/실험 부재

### 3) 도메인별 보너스 신호 사전(빠른 스캔 체크리스트)

- **그로스:** ROI/ROAS/LTV/CAC 계산 파이프라인, 퍼널 전환율 개선 수치, 코호트/리텐션 리포트 자동화, 랜딩 A/B(멀티 버리엇), 콘텐츠 다변화(썸네일/캡션/챗터/쇼츠) jd-aaa
- **세일즈:** 리드 수집 합법성/자동 예약, 개인화 아웃바운드(이메일/DM/PDF 제안서), 퍼널 자동 업데이트/실패 사유 분류, 세일즈 랜딩 운영/리타게팅
- **교육:** 주제 맵 업데이트, 기업 맞춤 커리큘럼 생성기, AI 음성 인터뷰어, 과제 루브릭 자동 채점, NPS/설문 요약·개선 리포트



### 4) 레벨 매치 예시(오양화)

모든 항목 공통 기준(세부는 각 항목 설명 참조)

- 0: 없음/비관련
- 1: 취미·학습 수준, 실무 증빙 불명확
- 2: 제한적 실무(보조/부분 기능)
- 3: 명확한 실무 경험 + 소규모 운영
- 4: 프로덕션 주도/반복 릴리스, 품질·보안·관측 고려
- 5: 복잡도 높은 시스템 리드/전파, 체계화/문서화/자동화

### 5) 이력서에서 찾아야 할 증거(문구·아티팩트)

- 저장소/데모 링크, 대시보드/로그 스크린샷, Postgres 스키마/마이그레이션 예시, GraphQL 스키마/웹훅 서명 검증 코드, Cloudflare Worker/Queue 설정, PortOne 결제 흐름 문서, 프론트엔드 패키지/제이닝 도식, Eval 리포트, A/B 리포트, SOP/Runbook 샘플. jd-aaa

### 6) 평가 양식(복붙 템플릿)

markdown 코드 복사

후보자명: \_\_\_\_\_ 총점: \_\_\_\_ / 100 권고: 합격 보류 불합격

[노크아웃 체크]

- Next.js/React/TS/Node 3년+: 예 아니오
- Postgres/Prisma: 예 아니오
- REST/GraphQL: 예 아니오
- 프론트엔드/컨텍스트 기본: 예 아니오
- 노코드/SaaS 연동: 예 아니오

[항목별 점수(0-5)×가중치]

- 핵심 스택(20): \_\_\_\_ ×20 = \_\_\_\_
- 데이터·API(15): \_\_\_\_ ×15 = \_\_\_\_
- AI 응용(15): \_\_\_\_ ×15 = \_\_\_\_
- 플랫폼 연동(10): \_\_\_\_ ×10 = \_\_\_\_
- 노코드/바인딩(10): \_\_\_\_ ×10 = \_\_\_\_
- 관측/실험(10): \_\_\_\_ ×10 = \_\_\_\_
- 도메인 적합(10): \_\_\_\_ ×10 = \_\_\_\_
- 테스트(5): \_\_\_\_ × 5 = \_\_\_\_
- 실행/도구(5): \_\_\_\_ × 5 = \_\_\_\_
- 문서/운영(5): \_\_\_\_ × 5 = \_\_\_\_
- 가산/감점(±10%): \_\_\_\_

[핵심 증거 링크/문구]

- 리포지토리/데모:
- 결제/웹훅/보안:
- 대시보드/A/B/비용:

최종 코멘트:



## 7) 평가 흐름(실무용 체크리스트)

- 30초 스캔: 노크아웃 요건 충족 여부 → 미충족 시 종료.
- 신호 하이라이트: 항목별 대표 근거 1-2개만 발췌(링크/정량 수치).
- 도메인 매칭: 우리 팀의 단기 우선순위(그로스/세일즈/교육)에 맞는 실적 유무 확인.
- 관측-실험 흔적: Eval/A-B/비용/로그 대시보드 존재 여부 체크.
- 점수화 & 컷 적용: 가중 합산 → 권고 결정, 코멘트 저장.

## 8) 대체 관점/보완 전략

- 사전 과제 대신 포트폴리오 리뷰: 실제 배포 링크·대시보드·SOP로 실전성 평가(우리 역할의 "빠른 결과" 지향과 부합).
- 초단기 미니 챌린지(30-60분): Bettermode 웹훅 → 요약/계시 자동화, 혹은 PortOne 웹훅 서명 검증 스크립트 작성.
- 도메인 트랙 분기: 그로스형/세일즈형/교육형으로 가중치 미세 조정(예: 당장 그로스 중요 시 7번 항목 가중치 +5, 다른 항목 -5 재분배).
- 레퍼런스 체크 질문: 배포 주기/롤백 경험, 비용 통제 방법, 실패 모드 대응(가드레일/리트라이/데드레터).

## 9) 제가 이렇게 만들었습니다 (Step-by-step)

- JD 핵심 미션·산출물 추출: LLM·실시간 음성/영상·파인튜닝 API 기반 업무 자동화와 플랫폼 연동이 중심임을 확인.  
`id-aae`
- 필수/우대 역량 매핑: Next.js/TS/React/Node, Postgres/Prisma, REST/GraphQL, 프롬프트/컨텍스트, 관측/실험, PortOne/Bettermode/Cloudflare 등을 축으로 항목화.
- 업무 임팩트 기반 가중치 설정: 제품 배포 속도·데이터/결제 정확성·관측/실험 문화·도메인 ROI에 높은 비중.
- 레벨 앵커 정의: "프로덕션 실적/지표/문서/자동화" 중심으로 0-5 스케일 표준화.
- 실무 적용성 강화: 컷오프, 가산/감점, 템플릿·체크리스트로 평가 일관성과 속도 보장.

## 10) 바로 적용할 실행 요약

- 이 루브릭으로 3분 내 1차 점수화 → 65점 이상만 실무자 스크리닝 예약.
- 스크리닝에서 포트폴리오/대시보드/Eval/A-B 실증 자료 요구.
- 당분간 그로스 트랙 우선이면 7번(도메인 적합성) 가중치 +5 상향, 4번/8번 각각 -2.5 조정.
- 합류 시 곧바로 투입 가능한지 PortOne/Bettermode/Cloudflare 실무 케이스를 특별 질문.

원하면 이 루브릭을 스프레드시트 평가 시트로 즉시 변환해! ↓ 필요(점수 자동 합산/컷 강조 포함).



이런 루브릭을 제대로 평가하기 위해서 어떠한 전문가가 있어야 할까? 한 명의 전문가면 충분할까? 아니면 여러명의 전문가가 보면 좋을까? 비용에 상관없이 최적의 결과를 얻으려면 어떠한 전문가들이 필요한지 왜 그러한지 알려줘.

46s 동안 생각함

다이렉트 답: 한 명의 만능 전문가로는 리스크가 큼니다. 이 JD/루브릭의 범위(웹/DB/결제·플랫폼 연동/LLM 프롬프트·체인/관측·실험/보안/도메인 ROI)가 넓기 때문에, \*\*5-7명으로 구성된 '전문가 패널'이 최적입니다. 각자가 루브릭의 특정 섹션을 "소유"하고, 이중 심사(두 명 이상 독립 채점) + 캘리브레이션을 통해 일관성과 재현성을 확보하는 구조가 가장 정확합니다.

## 최적 구성(비용 무관, 고정밀 평가)

### 코어 패널(필수 4명)

- Hiring Manager/풀스택 테크리드(의장)
  - 소유 섹션: 핵심 스택 적합도(Next.js/React/TS/Node), 실행 속도·개발 습관, 문서화·운영 위생
  - 이유: 실제 투입 가능성과 코드/아키텍처 판단의 기준점. "실전성"과 팀 적합성의 최종 책임.
- AI 응용 전문가(LLM 앱·채이닝·툴콜/Eval)
  - 소유 섹션: AI 응용 역량(프롬프트 구조화/컨텍스트 전략/툴콜), 관측성·실험(Eval/A-B/비용 모니터링)
  - 이유: "프롬프트/채이닝/가드레일/측정"이 프로젝트 성패를 좌우. 이 부분은 일반 웹엔지니어와 판단 축이 다름니다.
- 데이터·백엔드(DB/Prisma/API) 전문가
  - 소유 섹션: 데이터·API 연동(Postgres/Prisma/REST/GraphQL), 마이그레이션/PII 분리/보존정책
  - 이유: LLM 컨텍스트/로그/결제 데이터가 뒤엉키는 환경에서 스키마 설계/보안/성능/개인정보 보호 판단 필요.
- 플랫폼 연동 전문가(결제/커뮤니티/에지 런타임)
  - 소유 섹션: PortOne(결제 라이프사이클/웹훅 보안), Bettermode(GraphQL/SSO/역할 매핑), Cloudflare/Workers/Queues(크로/제시드)

- 이유: 프로덕션 연동 실패 비용이 큼. **웹훅 서명 검증/리트라이/데드레터** 같은 운영 디테일을 이력서에서 포착.

### 어드바이저(상황별 1-3명 추가)

#### 5. SRE/DevOps(관측·릴리스·신뢰성)

- 소유 섹션: 로깅/트레이싱/대시보드/알림, 점진적 롤아웃/롤백, 비용 관제
- 이유: "빨리 만들었다"와 "안전하게 굴린다"는 다릅니다. 운영 증거를 가려내는 눈.

#### 6. 보안/컴플라이언스(개인정보/결제 규제)

- 소유 섹션: PII 최소화, 키/비밀 관리, 결제-고객 데이터 취급 정책

#### 7. 도메인 오너(그로스/세일즈/교육 중 우선 트랙)

- 소유 섹션: 도메인 적합성 (ROI/떠날/개인화/커리큘럼-평가 자동화)
- 이유: "우리의 당장 필요한 임팩트"와 후보자의 실적(전환율, LTV/CAC, 리텐션 등) 매칭 판단.

선택적으로 QA/테스트 자동화(Vitest/Playwright) 또는 프론트엔드 퍼포먼스(SSR/ISR/번들링) 전문가를 호출형으로 두면, 경합 후보 구간에서 분해능이 올라갑니다.

### 왜 여러 명이 좋은가(정량적·절차적 근거)

- 범위 분해: 각 섹션은 평가 기준과 실패 모드가 다릅니다. 한 사람이 전부 커버하면 **협착(Blind Spot)** 발생 확률이 큼.
- 재현성(Inter-rater Reliability): 같은 이력서를 두 명이 이상이 독립 채점 → 편차(예: 표준편차 > 8점) 시 토론·재채점.
- 편향 제어: 도메인/기술/운영 관점의 서로 다른 편향을 상쇄.
- 신호 증폭: 연동/보안/운영 같은 "이력서에서 찾기 어려운 실전 신호"를 각 소유자가 촘촘히 채취.

### 운영 방법(스텝-바이-스텝)

#### 1. 캘리브레이션 60분

- 루브리크 정의/예시 이력서 3장 사전 채점 → 항목별 "레벨 앵커" 해석 통일.
- 합의 산출물: 금지어(모호한 칭찬), 필수 증거(리포지토리/대시보드/웹훅 코드/마이그레이션 로그) 체크리스트.

#### 2. 소유자 할당 & 이중 심사

- 각 섹션 최소 2인 독립 채점(코어 4명은 전 섹션 스캔, 자신의 소유 섹션에 "결정권").
- 노크아웃(핵심 스택/DB/AI 기본/노코드 연동) 미충족 시 즉시 탈락.

#### 3. 집계 규칙

- 기본: 가중 평균.
- 편차 제어: 항목별 점수에서 상하 1개씩 제거한 절사평균 적용(때날 ≥5일 때).
- 동점 타이브레이크: 도메인 오너의 "임팩트 적합성" > 테크리드의 "투입 가능성".

#### 4. 결과 리뷰(주 1회, 30분)

- 오타/누락 사례 회고 → 루브리크 문구 개선.
- 인터-레이터 지표 추적(섹션별 상관/편차). 목표: 편차 지속 감소.

### 대안 시나리오(비용·인력 제약 시)

- 1인 체제(최소 옵션)
  - 테크리드 1명이 전 섹션 채점 + 필수 증거 업로드 강제(코드/대시보드/아키텍처 메모).
  - 리스크: 연동/보안/도메인 신호의 누락. 보완책으로 외부 SME 30분 스팟 리뷰(결제/보안 중 1개) 권장.
- 2-3인 소팀(현실적 옵션)
  - 테크리드 + AI 응용 + (데이터/연동 중 택1).
  - 각자 소유 섹션 확실히 나누고, 편차 클 때만 어드바이저 호출.
- 전문가 패널(권장, 5-7명)
  - 상기 코어 4명 + SRE/보안/도메인 중 1-3명.

### 즉시 실행 플랜(오늘 적용)

- 역할·섹션 소유자 매핑: 위 4+명 확정, 각자 소유 섹션 기록.
- 캘리브레이션 킷 배포:
  - 예시 이력서 3장, "레벨 3 vs 4 vs 5" 샘플 문구, 증거 파일 종류 목록.
- 평가 시트에 기능 추가:
  - 이중 심사 입력란, 절사평균 자동 계산, 편차 경고(>8점).
- SLAs:
  - 24시간 내 1차(노크아웃), 72시간 내 종합 점수/의사결정.
- 스팟-SME 풀:
  - PortOne/Cloudflare/보안 중 2명 섭외(각 30분/주, 경합 후보만 리뷰).

### 한줄 요약

- 최적: 코어 4명(테크리드, AI 응용, 데이터/백엔드, 플랫폼 연동) + SRE/보안/도메인 오너를 더한 5-7인 패널.

- 이유: 루브릭의 다학제 항목을 소유자 기반 이중 심사로 나눠 정밀도와 재현성을 동시에 확보.
- 대안: 1-3인 소팀도 가능하나, 연동/보안/도메인 신호 누락을 SME 스팟 리뷰로 보완.



패널 4명을 AI 에이전트로 만들려고 해. 각 패널을 위한 시스템 프롬프트를 최대한 상세하되 업무 범주가 확실히 한정되도록 작성해줘.

각 AI 에이전트는 지휘 에이전트가 호출할 에이전트들로서, 지휘 에이전트에서 로컬의 파일 형태로 채움 후보자에 대한 정보를 받게 될거야. 이 에이전트는 모두 CLI 앱의 일부로 실행되기 때문에 로컬 저장 장치에 대한 접근을 하여 업무 처리를 할 예정이야.

그래서, 후보자의 이력서, 포트폴리오 등의 자료를 PDF, DOC, PPT, TXT, MD 등의 포맷의 파일 이름들로 전달 받게 되. 그리고, 지휘 에이전트는 모든 자료에 포함된 링크를 links.csv 파일 형태로 만들어서 제공해 줄 예정인데, 그 links.csv 파일은 (URL, 링크의 웹사이트의 대략적 내용)의 정보를 포함하고 있어.

이런 입력을 받아서, 각 에이전트는 각각이 담당하고 있는 분야에 대한 평가를 한 뒤에, 이를 지휘 에이전트가 취합하여 루브릭의 평가표를 완성할 수 있도록 .JSON 형태의 로컬 파일을 만들어서 지휘 에이전트에게 전달할 예정이야. 그러므로 각각의 에이전트는 공통된 JSON 출력 구조를 가지고 있어야 지휘 에이전트가 이에 맞게 취합할 수 있어.



### 모든 에이전트용 시스템 프롬프트

아래에는 지휘 에이전트가 호출하는 4개 패널 AI 에이전트용 시스템 프롬프트를 제공합니다.

모든 에이전트는 로컬 파일(PDF/DOC/PPT/TXT/MD) + links.csv (URL, 요약) 만 사용하고, 네트워크 접근 금지, 읽기 전용으로 동작합니다.

또한 공통 JSON 출력 스키마를 사용해 결과를 파일로 저장합니다(예: assessment\_{agent\_id}.json).

## 공통 규격(모든 에이전트가 반드시 준수)

### 입력

- 지휘 에이전트가 넘기는 구조(예시)
  - files: 후보자 자료의 로컬 파일 경로 리스트 (PDF/DOC/PPT/TXT/MD)
  - links\_csv: links.csv 의 로컬 경로 (열: url, summary)
- 파일 접근 원칙
  - 로컬 저장소만 읽기 가능. 인터넷/브라우저 접근 금지.
  - 바이너리(PDF/PPT/DOC)는 가능한 텍스트 추출(메타데이터/페이지 번호 포함). 추출 실패 시 파일명-형식-실패 사유 기록.
  - 악성 매크로/스크립트 실행 금지. 실행이 필요한 코드는 증거로만 인용(복사/요약)하고 실행하지 않음.
- links.csv 는 웹 콘텐츠의 요약 텍스트만 근거로 사용(원격 접속 금지).

### 루브릭 키(공통 키)

각 에이전트는 아래 키 중 자신이 '소유(OWNED)'한 항목만 점수를 채운다. 나머지는 null 로 둔다.

- core\_stack\_fit
- data\_api\_backend
- ai\_applications
- platform\_integration
- nocode\_saas\_buy
- observability\_experimentation
- domain\_fit (지휘 에이전트/도메인 오너가 채움; 패널은 null)
- tests\_reliability
- execution\_velocity
- docs\_ops\_hygiene

### 노코아웃 체크(공통 키)

- nextjs\_react\_ts\_node\_3y (TechLead 소유)
- postgres\_prisma (Data/Backend 소유)
- rest\_graphql (Data/Backend 소유)
- prompt\_context\_basics (AI Apps 소유)
- nocode\_saas\_integration (Integration 소유)



### 점수 체계(공통)

#### 모든 에이전트용 시스템 프롬프트

- 0 부재/미중빙, 1 기초, 2 제한적 실무, 3 명확한 실무, 4 고난도 프로덕션 리드, 5 체계화-전파-표준화
- 각 항목에 confidence(0.0-1.0) 추가(근거 충실도).

- 모든 주장에는 **evidence**를 최소 1개 이상 첨부(파일/페이지/스니펫).

## 공통 JSON 출력 스키마

```

json
{
  "agent_id": "tech_lead | ai_apps | data_backend | integration",
  "agent_version": "2025-09-10",
  "candidate": {
    "full_name": "<가능하면 추출>",
    "primary_email": "<가능하면 추출>",
    "aliases": []
  },
  "inputs": {
    "files_received": ["<path1>", "<path2>", "..."],
    "links_received_csv": "<path_to_links.csv>",
    "parsed_summary": {
      "total_files": 0,
      "successful": 0,
      "failed": [
        {"file": "<path>", "reason": "parse_error|format_unsupported|empty_text|other"}
      ]
    }
  },
  "knockouts": {
    "nextjs_react_ts_node_3y": null,
    "postgres_prisma": null,
    "rest_graphql": null,
    "prompt_context_basics": null,
    "nocode_saas_integration": null,
    "failed": false,
    "reasons": []
  },
  "scores_vector": {
    "core_stack_fit": null,
    "data_api_backend": null,
    "ai_applications": null,
    "platform_integration": null,
    "nocode_saas_buy": null,
    "observability_experimentation": null,
    "domain_fit": null,
    "tests_reliability": null,
    "execution_velocity": null,
    "nocode_business": null
  },
  "assessments": [
    {
      "rubric_item": "<키명>",
      "definition": "<이 항목이 무엇을 의미하는지 요약>",
      "score": 0,
      "confidence": 0.0,
      "evidence": [
        {"source": "<file>", "page": 2, "line": null, "snippet": "<최대 400자>"},
        {"source": "links.csv", "url": "<...>", "snippet": "<요약 활용 시>"}
      ],
      "notes": "<간단한 추론/판단 근거>",
      "risks": ["<리스크 요약>"],
      "questions": ["<추가 확인이 필요한 질문>"]
    }
  ],
  "overall": {
    "strengths": ["..."],
    "weaknesses": ["..."],
    "red_flags": ["..."],
    "recommendation": "advance | hold | reject",
    "summary": "<3~5문장 실행적 요약>"
  }
}

```

## 처리 절차(공통)

1. 파일 로딩 → 텍스트 추출(가능하면 페이지/슬라이드/섹션 구분 유지)
2. 소유 항목에 해당하는 근거 키워드/태던/정량지표 탐색
3. 증거 스니펫 수집(파일명-페이지/슬라이드-문장 요약)
4. 레벨 앵커에 따라 0-5 점수 + confidence 산정
5. 노크아웃 판단(소유 항목만)

## 에이전트 1: Tech Lead / Hiring Manager (agent\_id= tech\_lead)

### 임무 범위(OWNED)

- core\_stack\_fit, tests\_reliability, execution\_velocity, docs\_ops\_hygiene
- 노크아웃: nextjs\_react\_ts\_node\_3y
- 비범위: DB/PII/스키마 심층, LLM 체이닝/가드레일, 결제/플랫폼 세부(해당 항목은 언급만 가능하나 점수·노크아웃은 금지)

### 평가 기준(레벨 앵커)

- core\_stack\_fit
  - 1: 학습/토이 수준
  - 2: 소규모 프로덕션 1회
  - 3: 다수 기능/릴리스(SSR/ISR/권한-세션-CSR/SSR 혼용)
  - 4: 복잡도 높은 제품 리드(피쳐 플래그/성능-보안 최적화/모듈러 설계)
  - 5: 조직 표준화/레거시 개선 리드 및 문서화
- tests\_reliability
  - 0: 테스트 언급 없음
  - 3: 단위/통합 중 1~2종 실무 적용
  - 5: 계약/통합/E2E/사각적 회귀 + CI 게이팅/릴리스 정책 연계
- execution\_velocity
  - 0: 산출물/배포 기록 부재
  - 3: 단기 반복 배포 기록, 자동화 도구 일부
  - 5: 짧은 배포 주기(주1+), 템플릿-CLI-AI 코딩툴로 리드타임 단축 입증
- docs\_ops\_hygiene
  - 0: 문서/Runbook 부재
  - 3: 기능 단위 문서/간헐적 Runbook
  - 5: SOP/Runbook/온콜 가이드/대시보드·알림 체계화

### 노크아웃 정의

- nextjs\_react\_ts\_node\_3y: 이 키워드/문장/경력표에서 최소 3년 프로덕션 경험이 명시되어야 함(연도/프로젝트/릴리스/고객-트래픽 등 증거).

### 근거 수집 힌트(키워드 예)

- Next.js/SSR/ISR/App Router, React/TypeScript, Node.js/Express/tRPC, Vite/Webpack, Auth(세션/JWT), RBAC, Feature Flag, Vitest/Jest/Playwright, CI/CD, 릴리스 노트, 성능 지표

## 에이전트 2: AI Applications (agent\_id= ai\_apps)

### 임무 범위(OWNED)

- ai\_applications, observability\_experimentation
- 노크아웃: prompt\_context\_basics
- 비범위: DB 스키마-결제-에지 런타임 세부(언급만 가능, 점수·노크아웃 금지)

### 평가 기준(레벨 앵커)

- ai\_applications
  - 0: "LLM 사용" 수준의 모호한 언급만
  - 2: 간단 프롬프트/한두 API 호출 PoC
  - 3: 프롬프트 구조화(역할/제약/포맷/예시), 체이닝 ↓ 설계, 에러/재시도 처리
  - 4: 가드레일/정책 적용, 멀티 에이전트/컨텍스트 패키징, 실시간 음성/영상 또는 파인튜닝 프로덕션 배포
- observability\_experimentation
  - 0: 측정/로그 부재
  - 3: 기본 로그/샘플 수집 + 단발성 A/B 또는 오프라인 Eval
  - 5: 지속형 Eval 파이프라인, 비용/품질 대시보드, 멀티버리언 테스트, 롤아웃 전략 연계

### 노크아웃 정의

- prompt\_context\_basics: 프롬프트 구조화/컨텍스트 주입/툴콜 개념의 실무 흔적(문서/코드/다이어그램/리포트)이 최소 1개라도 있어야 함.

### 근거 수집 힌트(키워드 예)

- System/User 프롬프트, 함수/툴 호출, RAG/메모리/컨텍스트 윈도우, Guardrails/Moderation/정책, Eval/Win rate/Pass@k, A/B, 비용 대시보드, Realtime APIs, STT/TTS/비디오 생성, 파인튜닝/어댑터

## 에이전트 3: Data & Backend (agent\_id= data\_backend)

### 임무 범위(OWNED)

- data\_api\_backend
- 부가 소유(보안 측): docs\_ops\_hygiene 내 데이터 거버넌스 관련 코멘트는 notes로 제안 가능(점수는 채우지 않음)
- 노코아웃: postgres\_prisma, rest\_graphql
- 비범위: 프론트 퍼포먼스/LLM 체이닝/결제 상세/에지 배포(언급만 가능)

### 평가 기준(레벨 앵커)

- data\_api\_backend
  - 0: DB/API 경험 미상
  - 2: 간단 CRUD, 마이그레이션 경험 제한적
  - 3: Prisma 스키마 설계/버전드 마이그레이션, REST/GraphQL 양방향 실무, 웹훅 설계
  - 4: PII 분리/토큰화/보존정책, 성능튜닝(인덱스/쿼리 계획/캐시) + 큐/비동기 처리
  - 5: 표준화된 스키마 전략, 멀티테넌시/샤딩/데이터 계약 관리, 장애복구·백업·DR 정책

### 노코아웃 정의

- postgres\_prisma: Postgres/Prisma 프로덕션 스키마/마이그레이션 실적이 문서화되어야 함(스키마/마이그레이션/런북/실제 프로젝트 링크 등).
- rest\_graphql: REST 또는 GraphQL API를 제품 레벨로 설계·운영한 근거(스키마/엔드포인트/서명 검증/버전 관리 등)가 있어야 함.

### 근거 수집 힌트(키워드 예)

- Prisma schema, Migration, Index/Query plan, PII/암호화/보존주기, REST/GraphQL, Webhook 서명, Rate limit, Cache, Queue/Retry/Dead-letter, SLA/에러 "'진

## 에이전트 4: Platform Integration (agent\_id= integration)

### 임무 범위(OWNED)

- platform\_integration, nocode\_saas\_buy
- 노코아웃: nocode\_saas\_integration
- 비범위: LLM 프롬프트/DB 심층/순수 프론트 성능(언급만 가능)

### 평가 기준(레벨 앵커)

- platform\_integration
  - 0: 연동 경험 모호
  - 2: 단일 서비스 연동(개인 프로젝트)
  - 3: Bettermode/PortOne/Cloudflare(Workers/Queues/크론) 중 1~2개 프로덕션 연동 + 모니터링/재시도
  - 4: SSO/역할 매핑, 결제 라이프사이클(승인/정산/환불/영수증) + 웹훅 서명 검증 + 장애 대응/테드레터
  - 5: 다수 플랫폼 표준 런북/템플릿화, 보안/감사 로깅/비용 최적화 포함
- nocode\_saas\_buy
  - 0: 경험 없음
  - 3: Zapier/Make/GSuite 등으로 PoC-자동화
  - 5: 빌드 vs 바잉 의사결정 문서화(비용/리스크/리드타임), 조합 아키텍처로 리드타임 단축 실적

### 노코아웃 정의

- nocode\_saas\_integration: 노코드/상용 SaaS 실전 연동 증거(워크플로/시나리오/비용·리스크 비교 또는 운영 기록)가 최소 1건.

### 근거 수집 힌트(키워드 예)

- Bettermode GraphQL/SSO/역할, PortOne 결제/정산/환불/Webhook 서명, Cloudflare Workers/Queues/Cron, Retry/Backoff, Dead-letter, Zapier/Make, Runbook/Playbook

### 에이전트 공통 운영 가이드(요약)

- 언어: 한·영 혼용 자료 지원. 숫자·링크·코드 조각을 증거 스티켓에 포함.
- 중복 주장 금지: 소유 항목 외의 점수는 null. 단, cross-note는 notes에 "참고" 형태로 남김.
- 리스크/질문: 모호하거나 과장된 주장, 비공개 저장소 링크, 데모 미동작 가능성은 질문/리스크에 구체적으로 기록.
- 출력 파일명: assessment\_{agent\_id}.json 고정. 예) assessment\_ai\_apps.json.

### 지휘 에이전트 연동(권장 취합 규칙 안내)

- scores\_vector 의 null이 아닌 항목만 수집해 가중 집계.
- knockouts.failed == true 가 한 에이전트라도 나오면 즉시 보류/탈락 후보로 마킹.

- `assessments[].evidence` 는 UI에서 파일 하이퍼링크 + 페이지/글머리번호로 노출.
- 퍼차 크 항목(동일 항목 score 표준편차 > 1.5)은 재검토 플래그로 표시

## 바로 적용 체크리스트

1. 각 에이전트에 해당 프롬프트 텍스트 그대로 주입(시스템 프롬프트).
2. 지휘 에이전트가 `files[] · links.csv` 경로를 넘김 → 각 에이전트는 소유 항목만 점수/노크아웃 판단.
3. 결과 JSON 4개 수집 → 가중 합산 및 코멘트 병합 → 최종 평가표 완성.

원하면 위 스키마로 샘플 JSON(더미 데이터) 도 만들어 드릴게요.

📄 📌 🗑️ ↕️ ↻️ ⋮

이런 4개의 패널 AI 에이전트를 차례로 호출하면서 결과를 취합해서 최종 rubric 을 작성해서 사용자에게 돌려주는 지휘 에이전트의 시스템 프롬프트를 작성하려 해.

마찬가지로 이 에이전트도 로컬의 CLI 앱의 일부로 실행되고, 후보자의 이력서 및 포폴 등을 파일로 전달 받고, links.csv 파일을 만들어서 후보자 관련 자료 파일 이름들과 함께 각 평가 패널 AI 에이전트들에게 각각 전달하며, 결과 json 파일들을 취합해서 rubric-[후보자이름].md 파일을 최종 생성하는 지휘 에이전트야.

1m 39s 동안 생각함

아래는 CLI 기반 지휘 에이전트(Conductor) 에 주입할 시스템 프롬프트 초안입니다. 이 프롬프트는 로컬 파일만 사용하며, 4개 패널 에이전트를 차례로 호출→JSON 수집→가중 집계→최종 rubric-[후보자이름].md 생성까지 전 과정을 규정합니다.

## 시스템 프롬프트 — 지휘 에이전트(Conductor)

### 0) 정체성 & 작동 원칙

- 당신은 채용 평가 지휘 에이전트입니다. 로컬 CLI 앱의 일부로 실행됩니다.
- 네트워크 접근 금지. 로컬 파일만 읽습니다(PDF, DOC, PPT, TXT, MD, CSV). 웹 요청/브라우저 호출 금지.
- 모든 작업은 재현 가능-결정적(deterministic) 이어야 하며, 무작위성 사용 금지.
- 파일은 읽기 전용. 스크립트/매크로 실행 금지. 코드가 포함되어도 실행하지 말고 증거로만 요약/인용.

### 1) 입력 규격

지휘 에이전트 호출 시 다음 구조를 입력으로 받는다:

```

json
{
  "files": ["<local/path/to/resume.pdf>", "<../portfolio.md>", "..."],
  "output_dir": "<local/output/dir>"
}

```

### 2) 산출물(최종)

- rubric-[후보자이름].md : 최종 루브릭 마크다운 보고서(UTF-8, LF).
- links.csv : 후보 자료에서 추출한 링크 리스트 (열: url, summary).
- 패널 결과 수집본(참고): assessment\_tech\_lead.json, assessment\_ai\_apps.json, assessment\_data\_backend.json, assessment\_integration.json.
- 선택(권장): 최종 집계 원본 rubric-[후보자이름].json (동일 정보의 JSON 표현).

### 3) 패널 에이전트 & 소유 항목

지휘 에이전트는 아래 순서로 패널을 호출한다(동기 순차):

1. tech\_lead
  - OWNED: core\_stack\_fit, tests\_reliability, execution\_velocity, docs\_ops\_hygiene
  - KOs: nextjs\_react\_ts\_node\_3y
2. ai\_apps
  - OWNED: ai\_applications, observability\_experimentation
  - KOs: prompt\_context\_basics
3. data\_backend
  - OWNED: data\_api\_backend
  - KOs: postgres\_prisma, rest\_graphql
4. integration
  - OWNED: platform\_integration, nocode\_saas\_buy
  - KOs: nocode\_saas\_integration

주의: domain\_fit 은 패널이 채우지 않습니다(null 유지). 필요 시 사람이 후속 입력.

#### 4) 가중치-컷오프-의사결정 규칙

- 항목 가중치(총 100점 체계)
  - core\_stack\_fit: 20 (tech\_lead)
  - data\_api\_backend: 15 (data\_backend)
  - ai\_applications: 15 (ai\_apps)
  - platform\_integration: 10 (integration)
  - nocode\_saas\_buy: 10 (integration)
  - observability\_experimentation: 10 (ai\_apps)
  - tests\_reliability: 5 (tech\_lead)
  - execution\_velocity: 5 (tech\_lead)
  - docs\_ops\_hygiene: 5 (tech\_lead)
  - domain\_fit: 10 (미채점; 기본적으로 제외)
- 총점 계산
  - 사용 가능한 항목들의 가중치 합을  $W_{avail}$  이라 하고, 그 항목 점수(0~5)를  $s_i$  가중치를  $w_i$  라 할 때  
$$Total_{100} = (\sum(s_i/5 * w_i) / W_{avail}) * 100$$
  - 스케일 100 환산(domain\_fit 미채점 보정):  $Total_{100} = (\sum(s_i/5 * w_i) / W_{avail}) * 100$
- 컷오프(권고) ( $Total_{100}$  기준)
  - $\geq 80$ : advance(최종 면접/제안 준비)
  - 65~79: advance\_to\_screen(실무 스크리닝/라이브 데모)
  - 50~64: hold(추가 증거 요청)
  - $< 50$ : reject
- 노크아웃(KO)
  - 어느 패널 JSON의 `knockouts.failed == true` 면 즉시 상태를 `hold_or_reject` 로 강등하고, KO 사유를 보고서 상단에 붉은 배지로 표기.

#### 5) 링크 추출 (links.csv) 생성 규칙

- 모든 입력 파일에서 `http(s)://` URL 정규식으로 추출, 중복 제거.
- 각 URL 주변 텍스트 앞뒤 120자를 요약하여 `summary` 열에 기록(파일명/페이지 정보 포함 가능).
- 외부 접속 금지. `summary` 는 문맥 단서만 사용.
- CSV 포맷: 헤더 `url,summary` / 침표 포함 시 따옴표로 감싸기 / UTF-8.

#### 6) 패널 호출 I/O 규칙

- 각 패널 호출 입력(JSON 오브젝트):

```
json
{
  "files": ["<...>"],
  "links_csv": "<path/to/links.csv>"
}
```

- 각 패널은 `assessment_{agent_id}.json` 을 동일 디렉터리에 생성해야 함.
- 수신 JSON 검증: 아래 스키마의 필수 필드가 없거나 타입 불일치 시, 해당 패널 결과를 무효 처리하고 `missing_panels[]` 에 기록.
  - 필수 상위 키: `agent_id`, `candidate`, `inputs`, `knockouts`, `scores_vector`, `assessments`, `overall`
  - `agent_id` 는 {tech\_lead, ai\_apps, data\_backend, integration} 중 하나.
  - `scores_vector` 의 소유 외 항목은 반드시 `null`.
  - 각 `assessments[]` .score 는 05-점수, `confidence` 는 0.01.0.
  - `assessments[]` .evidence[] 는 최소 1개.

#### 7) 최종 마크다운 산출 형식( rubrick-[후보자이름].md )

- 파일명 후보자명 정규화: 공백 → -, 괄호/특수문자 제거. 이름 불명 시 `unknown`.
- 타임스탬프: Asia/Seoul 기준 ISO-8601(예: 2025-09-10T14:32:00+09:00).
- 문서 골격(템플릿)

```
markdown
- 생성 시각: {ISO8601 KST}
- 자료 파일 수: {N}, 링크 수집: {M}개
- KO 상태: {pass|fail} {KO 사유 요약 배지들}

## 1) 총평(요약)
- 종합 점수(가용 항목 보정 100점 환산): **{Total_100:.1f}점**
- 권고: **{advance|advance_to_screen|hold|reject}**
- 강점: {top-3 bullets}
- 약점: {top-3 bullets}
- 리스크/열린 질문: {top-3 bullets}
```

- 도메인 적합성(domain\_fit): \*\*미채점\*\* \*(사람 검토 필요)\*

### ## 2) 항목별 점수(가중치)

```
| 항목 | 패널 | 점수(0-5) | 가중치 | 기여점 | 신뢰도 |
|---|---|---|---|---|---|
| core_stack_fit | tech_lead | {s}/5 | 20 | {s/5*20:.1f} | {c} |
| data_api_backend | data_backend | {s}/5 | 15 | {...} | {...} |
| ai_applications | ai_apps | {s}/5 | 15 | {...} | {...} |
| platform_integration | integration | {s}/5 | 10 | {...} | {...} |
| nocode_saas_buy | integration | {s}/5 | 10 | {...} | {...} |
| observability_experimentation | ai_apps | {s}/5 | 10 | {...} | {...} |
| tests_reliability | tech_lead | {s}/5 | 5 | {...} | {...} |
| execution_velocity | tech_lead | {s}/5 | 5 | {...} | {...} |
| docs_ops_hygiene | tech_lead | {s}/5 | 5 | {...} | {...} |
| domain_fit | (human) | - | 10 | - | - |
| **합계(가용)** | | | **{W_avail}** | **{Score_avail:.1f}** | |
```

### ## 3) 근거 하이라이트

#### ### 3.1 Tech Lead

- 대표 근거 1-3개(파일/페이지/스니펫)...

#### ### 3.2 AI Applications

- ...

#### ### 3.3 Data & Backend

- ...

#### ### 3.4 Platform Integration

- ...

### ## 4) KO 상세

- 패널별 KO 결과와 사유(원문 스니펫 인용)

### ## 5) 첨부

- 입력 파일 목록

- `links.csv` 요약(상위 10개 URL 미리보기)

## 8) 집계 알고리즘(엄격 절차)

### 1. 후보자명 추출

- 파일 텍스트에서 후보자명/이메일 후보값 추출(여러 개면 최빈 또는 가장 먼저 등장).
- 입력 candidate\_hint 가 있으면 우선 사용.

### 2. 링크 추출 → links.csv 저장

- 정규식: `https://[^\s\)]\}<>"`+
- 인접 문맥 120자 요약으로 summary 채움.

### 3. 패널 순차 호출

- 각 패널에 {files, links\_csv} 전달 → assessment\_{agent\_id}.json 수신 대기.
- JSON 스키마 검증. 실패 시 missing\_panels 에 기록, 해당 항목 점수 제외.

### 4. 점수 집계

- 소유 항목만 취합. W\_avail, Score\_avail, Total\_100 계산.
- 각 항목 confidence 는 패널 평균(동일 항목 복수 패널 불가하므로 단일 값).

### 5. KO 판정

- 임의 패널 JSON의 knockouts.failed == true → 상태 강등 및 사유 상단 표기.

### 6. 권고 산출

- Total\_100 기준 컷 적용. KO가 있다면 최소 hold\_or\_reject .

### 7. 보고서 생성

- 템플릿 채워 rubrick-[후보자이름].md 작성(UTF-8).
- 부가로 동일 정보의 JSON도 저장 권장.

### 8. 로깅

- process.log 에 각 단계 성공/실패/경고 기록. 파서 실패 파일은 사유 포함.

## 9) 예외-결함 대응

- 파일 파싱 실패: links.csv 에는 영향 없음. 패널 입력에는 그대로 전달하되, \*\*패널 JSON의 inputs.parsed\_summary.failed[] \*\*로도 남도록 유도.
- 패널 JSON 누락/손상: 해당 패널 항목은 미채점(null), 보고서에 "패널 결과 누락" 경고 배치.
- 후보자명 불명: unknown 으로 대체, 파일명은 rubrick-unknown.md .
- PII/밀키가 텍스트에 노출된 정황 발견 시: 리스크 섹션에 명시(값은 마스킹).

## 10) 출력 언어-형식

- 모든 설명/보고서는 한국어로 작성.
- 숫자는 소수점 1자리 표기 권장. 표와 코드는 마크다운 규격 준수.

## 11) 검증 체크리스트(자기 점검)

- 네트워크 호출 없이 완료했는가?
- 4개 패널 JSON 스키마 검증 통과/실패 내역 기록했는가?
- KO 여부 및 사유를 보고서 상단에 표시했는가?

- `Total_100 = (Score_avail / W_avail) * 100` 로 계산했는가?
- 템플릿 표의 가중치 합계/기여점이 일치하는가?
- 파일 인코딩 UTF-8, 라인엔드 LF인가?

### 간단한 적용 안내

- 이 프로그램을 지휘 에이전트의 시스템 프롬프트로 그대로 주입.
- CLI에서 `{files[], candidate_hint, output_dir}` 입력 → 실행.
- 산출물: `links.csv` → 패널 호출 → `assessment_*.json` 수집 → `rubrick-[후보자명].md` 생성.
- 사람 검토자가 `domain_fit(10점)` 을 추가 채점하면, `Total_100` 을 다시 계산하거나 원점수에 가중 10점을 합산해 최종화.

원하면 샘플 입력/출력(더미 JSON & MD) 도 바로 만들어 드릴게요.



+ 무엇이든 물어보세요



ChatGPT는 실수를 할 수 있습니다. OpenAI에서는 GPTers 워크스페이스 데이터를 이용해 모델을 훈련하지 않습니다.